"Express Mail" Mailing Label No. **EL917901737US**

PATENT APPLICATION
ATTORNEY DOCKET NO. SUN-P7076-SPL

APPLYING TERM CONSISTENCY TO THE SOLUTION OF UNCONSTRAINED INTERVAL GLOBAL OPTIMIZATION PROBLEMS

Inventors: G. William Walster and Eldon R. Hansen

BACKGROUND

Field of the Invention

The present invention relates to performing arithmetic operations on interval operands within a computer system. More specifically, the present invention relates to a method and an apparatus for using a computer system to solve an unconstrained global optimization problem with interval arithmetic and term consistency.

Related Art

Rapid advances in computing technology make it possible to perform trillions of computational operations each second. This tremendous

computational speed makes it practical to perform computationally intensive tasks as diverse as predicting the weather and optimizing the design of an aircraft engine. Such computational tasks are typically performed using machine-representable floating-point numbers to approximate values of real numbers. (For example, see the Institute of Electrical and Electronics Engineers (IEEE) standard 754 for binary floating-point numbers.)

In spite of their limitations, floating-point numbers are generally used to perform most computational tasks.

One limitation is that machine-representable floating-point numbers have a fixed-size word length, which limits their accuracy. Note that a floating-point number is typically encoded using a 32, 64 or 128-bit binary number, which means that there are only 2^{32}, 2^{64} or 2^{128} possible symbols that can be used to specify a floating-point number. Hence, most real number values can only be approximated with a corresponding floating-point number. This creates estimation errors that can be magnified through even a few computations, thereby adversely affecting the accuracy of a computation.

A related limitation is that floating-point numbers contain no information about their accuracy. Most measured data values include some amount of error that arises from the measurement process itself. This error can often be quantified as an accuracy parameter, which can subsequently be used to determine the accuracy of a computation. However, floating-point numbers are not designed to keep track of accuracy information, whether from input data measurement errors or machine rounding errors. Hence, it is not possible to determine the accuracy of a computation by merely examining the floating-point number that results from the computation.

Interval arithmetic has been developed to solve the above-described problems. Interval arithmetic represents numbers as intervals specified by a first

(left) endpoint and a second (right) endpoint. For example, the interval $[a, b]$, where $a < b$, is a closed, bounded subset of the real numbers, R, which includes a and b as well as all real numbers between a and b. Arithmetic operations on interval operands (interval arithmetic) are defined so that interval results always contain the entire set of possible values. The result is a mathematical system for rigorously bounding numerical errors from all sources, including measurement data errors, machine rounding errors and their interactions. (Note that the first endpoint normally contains the "infimum", which is the largest number that is less than or equal to each of a given set of real numbers. Similarly, the second endpoint normally contains the "supremum", which is the smallest number that is greater than or equal to each of the given set of real numbers.)

One commonly performed computational operation is to perform global optimization to find a global minimum of an objective function. This can be accomplished using any members of a set of criteria to delete boxes, or parts of boxes that cannot contain the global minimum f^*. The set of criteria includes:

(1) the *f_bar*-criterion, wherein if *f_bar* is the least upper bound so far computed on f, then any point $\mathbf{x}$ for which $f(\mathbf{x}) > f\_bar$ can be deleted. Similarly, any box $\mathbf{X}$ can be deleted if $inf(f(\mathbf{X})) > f\_bar$;

(2) the monotonicity criterion, wherein if $\mathbf{g}(\mathbf{x})$ is the gradient of f evaluated at $\mathbf{x}$, then any point $\mathbf{x}$ for which $\mathbf{g}(\mathbf{x}) \neq \mathbf{0}$ can be deleted. Similarly, any box $\mathbf{X}$ can be deleted if $0 \notin \mathbf{g}(\mathbf{X})$;

(3) the convexity criterion, wherein if $H_{ii}(\mathbf{x})$ is the i-th diagonal element of the Hessian of f, then any point $\mathbf{x}$ for which $H_{ii}(\mathbf{x}) \geq 0$ (for $i=1,\ldots,n$) can be deleted. Similarly, any box $\mathbf{X}$ can be deleted if $H_{ii}(\mathbf{X}) \geq 0$ (for $i=1,\ldots,n$); and

(4) the stationary point criterion, wherein any point $\mathbf{x}$ is deleted using the interval Newton technique to solve the equation $\mathbf{g}(\mathbf{x}) \geq \mathbf{0}$.

All of these criteria work best "in the small" when the objective function f is approximately quadratic. For large intervals containing multiple local minima and maxima none of the criteria will succeed in deleting much of a given box. In this case the box is split into two or more sub-boxes , that are then processed independently. By this mechanism *all* the global minima of a nonlinear objective function can be found.

One problem is applying this procedure to large n-dimensional interval vectors (or boxes) that contain multiple local minima. In this case, the process of splitting in n-dimensions can lead to exponential growth in the number of boxes to process.

It is well known that this problem (and even the problem of computing "sharp" bounds on the range of a function of n-variables over an n-dimensional box) is an "NP-hard" problem. In general, NP-hard problems require an exponentially increasing amount of work to solve as n, the number of independent variables, increases.

Because NP-hardness is a worst-case property and because many practical engineering and scientific problems have relatively simple structure, one problem is to use this simple structure of real problems to improve the efficiency of interval global optimization algorithms.

Hence, what is needed is a method and an apparatus for using the structure of a nonlinear objective function to improve the efficiency of interval global optimization software. To this end, what is needed is a method and apparatus that efficiently deletes "large" boxes or parts of large boxes that the above criteria can only split.

SUMMARY

One embodiment of the present invention provides a system that solves an unconstrained interval global optimization problem specified by a function f, wherein f is a scalar function of a vector $\mathbf{x} = (x_1, x_2, x_3, \ldots x_n)$. The system operates by receiving a representation of the function f, and then performing an interval global optimization process to compute guaranteed bounds on a globally minimum value f^* of the function $f(\mathbf{x})$ and the location or locations x^* of the global minimum. While performing the interval global optimization process, the system deletes all of part of a subbox $\mathbf{X}$ for which $f(\mathbf{x}) > f\_bar$, wherein f_bar is the least upper bound on f^* that has been so far found. This is called the "f_bar test". The system applies term consistency to the f_bar test over the subbox $\mathbf{X}$ to increase that portion of the subbox $\mathbf{X}$ that can be proved to violate the f_bar test.

For any function of n-variables $f(\mathbf{x})$ there are different ways to analytically solve for a component x_j of the vector $\mathbf{x}$. For example, one can write $f(\mathbf{x}) \geq \mathbf{g}(x_j) - h(\mathbf{x})$, where $\mathbf{g}(x_j)$ is a term in f for which it is possible to solve $\mathbf{g}(x_j) = \mathbf{y}$ for any x_j using $g^{-1}(\mathbf{y})$. For each of these rearrangements, if a given interval box $\mathbf{X}$ is used as an argument of h, then the new interval X_j^+ for the j-th component of $\mathbf{X}$, is guaranteed to be at least as narrow as the original, X_j .

$$X_j^+ = X_j \cap X'_j \text{ where } X'_j = g^{-1}(h(\mathbf{X})).$$

This process is then be iterated using different terms g of the function f. After reducing any element X_j of the box $\mathbf{X}$ to X_j^+, the reduced value can be used in $\mathbf{X}$ thereafter to speed up the reduction process using other component functions and terms thereof.

Hereafter, the notation $g(x_j)$ for a term of the function $f(\mathbf{x})$ implicitly represents any term of any component function. This eliminates the need for additional subscripts that do not add clarity to the exposition.

In one embodiment of the present invention, applying term consistency involves symbolically manipulating an equation within the computer system to solve for a first term, $g(\mathbf{x})$, thereby producing a modified equation $g(\mathbf{x}) = h(\mathbf{x})$, wherein the first term $g(\mathbf{x})$ can be analytically inverted to produce an inverse function $g^{-1}(\mathbf{y})$. Next, the system substitutes the subbox $\mathbf{X}$ into the modified equation to produce the equation $g(\mathbf{X}') = h(\mathbf{X})$ and then solves for $\mathbf{X}' = g^{-1}(h(\mathbf{X}))$. The system then intersects $\mathbf{X}'$ with the subbox $\mathbf{X}$ to produce a new subbox $\mathbf{X}^+$, which contains all solutions of the equation within the subbox $\mathbf{X}$, and wherein the size of the new subbox $\mathbf{X}^+$ is less than or equal to the size of the subbox $\mathbf{X}$.

In one embodiment of the present invention, while performing the interval global optimization process, the system keeps track of a least upper bound *f_bar* on the global minimum f^*, and removes from consideration any subbox for which $f(\mathbf{X}) > f\_bar$. The system also applies term consistency to the inequality $f(\mathbf{X}) \leq f\_bar$, and excludes any portion of the subbox $\mathbf{X}$ that can be proved to violate the inequality.

In one embodiment of the present invention, while performing the interval global optimization process, the system determines a gradient $\boldsymbol{g}(\mathbf{x})$ of the function $f(\mathbf{x})$, wherein $\boldsymbol{g}(\mathbf{x})$ includes components $g_i(\mathbf{x})$ $(i=1,\ldots,n)$. The system then removes from consideration any subbox for which $\boldsymbol{g}(\mathbf{x})$ is bounded away from zero, thereby indicating that the subbox does not include a global minimum of $f(\mathbf{x})$. The system also applies term consistency to each component $g_i(\mathbf{x})=0$ $(i=1,\ldots,n)$ of $\boldsymbol{g}(\mathbf{x})=0$ over the subbox $\mathbf{X}$, and excludes any portion of the subbox $\mathbf{X}$ that can be proved to violate a component.

In one embodiment of the present invention, while performing the interval global optimization process, the system determines diagonal elements $H_{ii}(\mathbf{x})$ $(i=1,\ldots,n)$ of the Hessian of the function $f(\mathbf{x})$, and removes from consideration any subbox for which a diagonal element of the Hessian is always negative, which

indicates that the f is not convex and consequently does not contain a global minimum within the subbox. In doing so, the system applies term consistency to each relation $H_u(\mathbf{x}) \geq 0$ $(i=1,...,n)$ over the subbox **X**, and excludes any portion of the subbox **X** that can be proved to violate these inequalities.

In one embodiment of the present invention, while performing the interval global optimization process, the system solves for stationary points for which $\mathbf{g}(\mathbf{x})=0$. Term consistency is applied to the solution of the system of equations $\mathbf{g}(\mathbf{x})=0$, just as is done for the solution of any set of nonlinear equations. For example, the system computes the Jacobian $\mathbf{J}(\mathbf{x},\mathbf{X})$ of the function f evaluated as a function of a point **x** over a subbox **X**. The system also computes an approximate inverse **B** of the center of $\mathbf{J}(\mathbf{x},\mathbf{X})$, and uses the approximate inverse **B** to analytically determine the system $\mathbf{Bg}(\mathbf{x})$. The system also applies term consistency to each component $(\mathbf{Bg}(\mathbf{x}))_i = 0$ $(i=1,...,n)$ for each variable x_i $(i=1,...,n)$ over the subbox **X**, and excludes any portion of the subbox **X** that can be proved to violate the equalities $(\mathbf{Bg}(\mathbf{x}))_i = 0$.

In one embodiment of the present invention, the system terminates attempts to further reduce the subbox **X** when the width of **X** is less than a first threshold value, and the magnitude of $f(\mathbf{X})$ is less than a second threshold value.

BRIEF DESCRIPTION OF THE FIGURES

FIG. 1 illustrates a computer system in accordance with an embodiment of the present invention.

FIG. 2 illustrates the process of compiling and using code for interval computations in accordance with an embodiment of the present invention.

FIG. 3 illustrates an arithmetic unit for interval computations in accordance with an embodiment of the present invention.

FIG. 4 is a flow chart illustrating the process of performing an interval computation in accordance with an embodiment of the present invention.

FIG. 5 illustrates four different interval operations in accordance with an embodiment of the present invention.

FIG. 6 is a flow chart illustrating the process of finding an interval solution to a nonlinear equation using term consistency in accordance with an embodiment of the present invention.

FIG. 7A is a first portion of a flow chart illustrating the process of using term consistency to solve an unconstrained interval global optimization problem in accordance with an embodiment of the present invention.

FIG. 7B is a second portion of a flow chart illustrating the process of using term consistency to solve an unconstrained interval global optimization problem in accordance with an embodiment of the present invention.

FIG. 7C is a third portion of a flow chart illustrating the process of using term consistency to solve an unconstrained interval global optimization problem in accordance with an embodiment of the present invention.

FIG. 7D is a fourth portion of a flow chart illustrating the process of using term consistency to solve an unconstrained interval global optimization problem in accordance with an embodiment of the present invention.

DETAILED DESCRIPTION

The following description is presented to enable any person skilled in the art to make and use the invention, and is provided in the context of a particular application and its requirements. Various modifications to the disclosed embodiments will be readily apparent to those skilled in the art, and the general principles defined herein may be applied to other embodiments and applications without departing from the spirit and scope of the present invention. Thus, the

present invention is not limited to the embodiments shown, but is to be accorded the widest scope consistent with the principles and features disclosed herein.

The data structures and code described in this detailed description are typically stored on a computer readable storage medium, which may be any device or medium that can store code and/or data for use by a computer system. This includes, but is not limited to, magnetic and optical storage devices such as disk drives, magnetic tape, CDs (compact discs) and DVDs (digital versatile discs or digital video discs), and computer instruction signals embodied in a transmission medium (with or without a carrier wave upon which the signals are modulated). For example, the transmission medium may include a communications network, such as the Internet.

Computer System

FIG. 1 illustrates a computer system 100 in accordance with an embodiment of the present invention. As illustrated in FIG. 1, computer system 100 includes processor 102, which is coupled to a memory 112 and a to peripheral bus 110 through bridge 106. Bridge 106 can generally include any type of circuitry for coupling components of computer system 100 together.

Processor 102 can include any type of processor, including, but not limited to, a microprocessor, a mainframe computer, a digital signal processor, a personal organizer, a device controller and a computational engine within an appliance. Processor 102 includes an arithmetic unit 104, which is capable of performing computational operations using floating-point numbers.

Processor 102 communicates with storage device 108 through bridge 106 and peripheral bus 110. Storage device 108 can include any type of non-volatile storage device that can be coupled to a computer system. This includes, but is not

limited to, magnetic, optical, and magneto-optical storage devices, as well as storage devices based on flash memory and/or battery-backed up memory.

Processor 102 communicates with memory 112 through bridge 106. Memory 112 can include any type of memory that can store code and data for execution by processor 102. As illustrated in FIG. 1, memory 112 contains computational code for intervals 114. Computational code 114 contains instructions for the interval operations to be performed on individual operands, or interval values 115, which are also stored within memory 112. This computational code 114 and these interval values 115 are described in more detail below with reference to FIGs. 2-5.

Note that although the present invention is described in the context of computer system 100 illustrated in FIG. 1, the present invention can generally operate on any type of computing device that can perform computations involving floating-point numbers. Hence, the present invention is not limited to the computer system 100 illustrated in FIG. 1.

Compiling and Using Interval Code

FIG. 2 illustrates the process of compiling and using code for interval computations in accordance with an embodiment of the present invention. The system starts with source code 202, which specifies a number of computational operations involving intervals. Source code 202 passes through compiler 204, which converts source code 202 into executable code form 206 for interval computations. Processor 102 retrieves executable code 206 and uses it to control the operation of arithmetic unit 104.

Processor 102 also retrieves interval values 115 from memory 112 and passes these interval values 115 through arithmetic unit 104 to produce results 212. Results 212 can also include interval values.

Note that the term "compilation" as used in this specification is to be construed broadly to include pre-compilation and just-in-time compilation, as well as use of an interpreter that interprets instructions at run-time. Hence, the term "compiler" as used in the specification and the claims refers to pre-compilers, just-in-time compilers and interpreters.

Arithmetic Unit for Intervals

FIG. 3 illustrates arithmetic unit 104 for interval computations in more detail accordance with an embodiment of the present invention. Details regarding the construction of such an arithmetic unit are well known in the art. For example, see U.S. Patent Nos. 5,687,106 and 6,044,454. Arithmetic unit 104 receives intervals 302 and 312 as inputs and produces interval 322 as an output.

In the embodiment illustrated in FIG. 3, interval 302 includes a first floating-point number 304 representing a first endpoint of interval 302, and a second floating-point number 306 representing a second endpoint of interval 302. Similarly, interval 312 includes a first floating-point number 314 representing a first endpoint of interval 312, and a second floating-point number 316 representing a second endpoint of interval 312. Also, the resulting interval 322 includes a first floating-point number 324 representing a first endpoint of interval 322, and a second floating-point number 326 representing a second endpoint of interval 322.

Note that arithmetic unit 104 includes circuitry for performing the interval operations that are outlined in FIG. 5. This circuitry enables the interval operations to be performed efficiently.

However, note that the present invention can also be applied to computing devices that do not include special-purpose hardware for performing interval operations. In such computing devices, compiler 204 converts interval operations

into a executable code that can be executed using standard computational hardware that is not specially designed for interval operations.

FIG. 4 is a flow chart illustrating the process of performing an interval computation in accordance with an embodiment of the present invention. The system starts by receiving a representation of an interval, such as first floating-point number 304 and second floating-point number 306 (step 402). Next, the system performs an arithmetic operation using the representation of the interval to produce a result (step 404). The possibilities for this arithmetic operation are described in more detail below with reference to FIG. 5.

Interval Operations

FIG. 5 illustrates four different interval operations in accordance with an embodiment of the present invention. These interval operations operate on the intervals X and Y. The interval X includes two endpoints,

$\underline{x}$ denotes the lower bound of X, and
$\overline{x}$ denotes the upper bound of X.

The interval X is a closed subset of the extended (including $-\infty$ and $+\infty$) real numbers R^* (see line 1 of FIG. 5). Similarly the interval Y also has two endpoints and is a closed subset of the extended real numbers R^* (see line 2 of FIG. 5).

Note that an interval is a point or degenerate interval if $X = [x, x]$. Also note that the left endpoint of an interior interval is always less than or equal to the right endpoint. The set of extended real numbers, R^* is the set of real numbers, R, extended with the two ideal points negative infinity and positive infinity:

$$R^* = R \cup \{-\infty\} \cup \{+\infty\}.$$

In the equations that appear in FIG. 5, the up arrows and down arrows indicate the direction of rounding in the next and subsequent operations. Directed rounding (up or down) is applied if the result of a floating-point operation is not machine-representable.

The addition operation $X + Y$ adds the left endpoint of X to the left endpoint of Y and rounds down to the nearest floating-point number to produce a resulting left endpoint, and adds the right endpoint of X to the right endpoint of Y and rounds up to the nearest floating-point number to produce a resulting right endpoint.

Similarly, the subtraction operation $X - Y$ subtracts the right endpoint of Y from the left endpoint of X and rounds down to produce a resulting left endpoint, and subtracts the left endpoint of Y from the right endpoint of X and rounds up to produce a resulting right endpoint.

The multiplication operation selects the minimum value of four different terms (rounded down) to produce the resulting left endpoint. These terms are: the left endpoint of X multiplied by the left endpoint of Y; the left endpoint of X multiplied by the right endpoint of Y; the right endpoint of X multiplied by the left endpoint of Y; and the right endpoint of X multiplied by the right endpoint of Y. This multiplication operation additionally selects the maximum of the same four terms (rounded up) to produce the resulting right endpoint.

Similarly, the division operation selects the minimum of four different terms (rounded down) to produce the resulting left endpoint. These terms are: the left endpoint of X divided by the left endpoint of Y; the left endpoint of X divided by the right endpoint of Y; the right endpoint of X divided by the left endpoint of Y; and the right endpoint of X divided by the right endpoint of Y. This division

operation additionally selects the maximum of the same four terms (rounded up) to produce the resulting right endpoint. For the special case where the interval Y includes zero, X/Y is an exterior interval that is nevertheless contained in the interval R^*.

Note that the result of any of these interval operations is the empty interval if either of the intervals, X or Y, are the empty interval. Also note, that in one embodiment of the present invention, extended interval operations never cause undefined outcomes, which are referred to as "exceptions" in the IEEE 754 standard.

Term Consistency

FIG. 6 is a flow chart illustrating the process of solving a nonlinear equation through interval arithmetic and term consistency in accordance with an embodiment of the present invention. The system starts by receiving a representation of a nonlinear equation $f(x) = 0$ (step 602), as well as a representation of an initial interval X (step 604). Next, the system symbolically manipulates the equation $f(x) = g(x') - h(x) = 0$ to solve for a term $g(x') = h(x)$, wherein the term $g(x')$ can be analytically inverted to produce and inverse function $g^{-1}(y)$(step 606).

Next, the system substitutes the initial interval X into $h(x)$ to produce the equation $g(X') = h(X)$ (step 608). The system then solves for $X' = g^{-1}(h(X))$ (step 610). The resulting interval X' is then intersected with the initial interval X to produce a new interval X^+ (step 612).

At this point, the system can terminate. Otherwise, the system can perform further processing. This further processing involves setting $X = X^+$ (step 614) and then either returning to step 606 for another iteration of term

consistency on another term g of $f(x)$, or by performing an interval Newton step on $f(x)$ and X to produce a new interval X^+ (step 616).

Examples of Applying Term Consistency

For example, suppose $f(x) - x^2 - x + 6$. We can define $g(x) = x^2$ and $h(x) = x-6$. Let $X = [-10,10]$. The procedural step is $(X')^2 = X - 6 = [-16,4]$. Since $(X')^2$ must be non-negative, we replace this interval by $[0,4]$. Solving for X', we obtain $X' = \pm [0,2]$. In replacing the range of $h(x)$ (i.e., $[-16,4]$) by non-negative values, we have excluded that part of the range $h(x)$ that is not in the domain of $g(x) = x^2$.

Suppose that we reverse the roles of g and h and use the iterative step $h(X') = g(X)$. That is $X' - 6 = X^2$. We obtain $X' = [6,106]$. Intersecting this result with the interval $[-10,10]$, of interest, we obtain $[6,10]$. This interval excludes the set of values for which the range of $g(X)$ is not in the intersection of the domain of $h(X)$ with X.

Combining these results, we conclude that any solution of $f(X) = g(X) - h(X) = 0$ that occurs in $X = [-10,10]$ must be in both $[-2,2]$ and $[6,10]$. Since these intervals are disjoint, there can be no solution in $[-10,10]$.

In practice, if we already reduced the interval from $[-10,10]$ to $[-2,2]$ by solving for g, we use the narrower interval as input when solving for h.

This example illustrates the fact that it can be advantageous to solve a given equation for more than one of its terms. The order in which terms are chosen affects the efficiency. Unfortunately, it is not known how best to choose the best order.

Also note that there can be many choices for $g(x)$. For example, suppose we use term consistency to narrow the interval bound X on a solution of $f(x) = ax^4 + bx + c = 0$. We can let $g(x) = bx$ and compute $X' = -(aX^4 + c)/b$ or

we can let $g(x) = ax^4$ and compute $X' = \pm [-(bX+c)/a]^{1/4}$. We can also separate x^4 into $x^2 * x^2$ and solve for one of the factors $X' = \pm [-(bX+c)/(aX^2)]^{1/2}$.

In the multidimensional case, we may solve for a term involving more than one variable. We then have a two-stage process. For example, suppose we solve for the term $1/(x+y)$ from the function $f(x,y) = 1/(x+y) - h(x,y) = 0$. Let $x \in X = [1,2]$ and $y \in Y = [0.5,2]$. Suppose we find that $h(X,Y) = [0.5,1]$. Then $1/(x+y) \in [0.5,1]$ so $x+y \in [1,2]$. Now we replace y by $Y = [0.5,2]$ and obtain the bound $[-1,1.5]$ on X. Intersecting this interval with the given bound $X = [1,2]$ on x, we obtain the new bound $X' = [1,1.5]$.

We can use X' to get a new bound on h; but his may require extensive computing if h is a complicated function; so suppose we do not. Suppose that we do, however, use this bound on our intermediate result $x + y = [1,2]$. Solving for y as $[1,2] - X'$, we obtain the bound $[-0.5,1]$. Intersecting this interval with Y, we obtain the new bound $Y' = [0.5,1]$ on y. Thus, we improve the bounds on both x and y by solving for a single term of f.

The point of these examples is to show that term consistency can be used in many ways both alone and in combination with the interval Newton algorithm to improve the efficiency with which roots of a single nonlinear equation can be computed. The same is true for systems of nonlinear equations.

Term Consistency and Unconstrained Interval Global Optimization

FIGs. 7A-7D collectively present a flow chart illustrating the process of using term consistency in solving an unconstrained global optimization problem in accordance with an embodiment of the present invention. This global optimization process computes guaranteed bounds on the globally minimum value f^* of the objective function $f(\mathbf{x})$ and guaranteed bounds on the point(s) x^* where $f(\mathbf{x})$ is a global minimum.

Assume a given box (or boxes) in which the solution is sought is placed in a list L_I of boxes to be processed. Set w_R=0. If a single box $\mathbf{X}^{(0)}$ is given, set $w_I = w(\mathbf{X}^{(0)})$. If more than one box is given, the system sets w^I equal to the width of the largest box.

If an upper bound on the minimum value of $f(\mathbf{x})$ is known, set *f_bar* equal to this value. Otherwise, the system sets *f_bar* = $+\infty$. Note that *f_bar* is the least upper bound encountered so far on $f(\mathbf{x})$, wherein $\mathbf{x}$ is a point in the initial box. Moreover, *f_bar* can be used to eliminate from consideration any subbox whose lower bound is greater than *f_bar*, because such a subbox cannot contain the globally minimum value f^*. Note that we can use the inequality $f(\mathbf{x}) \geq$ *f_bar* to delete parts of a subbox and can use term consistency in the process.

A box size tolerance ε_X and a function width tolerance ε_F are specified to facilitate termination when the box size and the function width become smaller than pre-specified values. This prevents unnecessary processing.

We let N^H denote the number of (consecutive) times that application of term consistency to the set of relations $H_{ii}(\mathbf{x}) \geq 0$ (i=1,...,n) fails to reduce the current box. (Note that $H_{ii}(\mathbf{x})$ (i=1,...,n) represent the diagonal elements of the Hessian of the objection function $f(\mathbf{x})$.) We begin with $N^H = 0$.

Thus, to initialize our process, we specify ε_X, ε_F, w_R, w_I, N^H, and the initial box(es). In addition, we specify a bound on *f_bar* if one is known.

We perform the following steps in the order given except as indicated by branching. For each step, the current box is denoted by $\mathbf{X}$ even though it may be altered in one or more steps.

The system first evaluates f at (an approximation for) the center of each initial box in L_I and uses the result to update *f_bar* if necessary (step 701). Note that *f_bar* is updated whenever a new upper bound on f over a subbox $\mathbf{X}$ is found to be lower than the current *f_bar*.

Next, for each initial box $\mathbf{X}$ in L_1, the system evaluates $f(\mathbf{X})$. The system then deletes any box for which the lower bound of $f(\mathbf{X})$ is greater than *f_bar* (step 702).

If L_1 is empty, the system goes to step 738. Otherwise, the system finds the box $\mathbf{X}$ in L_1 for which the infimum of $f(\mathbf{X})$ is smallest. The system selects this box as the one to be processed next. For later reference denote this box by $\mathbf{X}^{(1)}$. The system also deletes $\mathbf{X}^{(1)}$ from L_1 (step 703).

Next, the system evaluates f at (an approximation for) the center of $\mathbf{X}$ and uses the result to update *f_bar*. The system skips this step if it has already been done for this box in step 701 (step 704).

If term consistency has been applied n times in step 706 to the relation $f(\mathbf{x}) \leq$ *f_bar* without having done step 710, the system goes to step 709 (step 705). (Note that the integer n is the number of variables upon which f depends.)

Next, the system applies term consistency to the relation $f(\mathbf{x}) \leq$ *f_bar*. If the result is empty, the system deletes $\mathbf{X}$ and goes to step 703 (step 706).

If $w(\mathbf{X}) < \varepsilon_X$ and $w(f(\mathbf{X})) < \varepsilon_F$, the system puts $\mathbf{X}$ in list L_2 and goes to step 703 (step 707).

If the box was sufficiently reduced in step 706, the system puts the result in L_1 and goes to step 703 (step 708). We say that a box $\mathbf{X}$ is sufficiently reduced if any component of $\mathbf{X}$ is reduced by an amount that is at least a fraction (say 0.25) of the width of the widest component of $\mathbf{X}$.

If term consistency has been applied n times in step 710 to the components of the gradient without having applied a Newton step in step 723, the system goes to step 722 (step 709).

Next, the system applies term consistency to $g_i(\mathbf{x})=0$ $(i=1,\ldots,n)$ for each component $g_i(\mathrm{x})$ of the gradient of $f(\mathbf{x})$. In so doing, the system bounds g_i over the

resulting box. If a result is empty, the system deletes **X** and goes to step 703 (step 710).

Next, the system uses the center of the bounds obtained from working on g_i in step 710 to do a line search and update *f_bar* (step 711). A line search can be performed as follows. Suppose we evaluate the gradient $\mathbf{g}(\mathbf{x})$ of $f(\mathbf{x})$ at a point **x**. Note that f decreases (locally) in the negative gradient direction from **x**. A simple procedure for finding a point where f is small is to search along this half-line. Let **x** be the center of the current box. Define the half-line of points $y(\alpha)=\mathbf{x}-\alpha\mathbf{g}(\mathbf{x})$ where $\alpha \geq 0$. We now use a standard procedure for finding an approximate minimum of the objective function f on this half-line. We first restrict our region of search by determining the value α' such that $y(\alpha')=\mathbf{x}-\alpha' g$ is on the boundary of the current box **X**, and search between **x** and $\mathbf{x}'=y(\alpha')$. We use the following procedure. Each application of the procedure requires an evaluation of f. Procedure: If $f(\mathbf{x}')<f(\mathbf{x})$, replace **x** by $(\mathbf{x}+\mathbf{x}')/2$. Otherwise, we replace **x'** by $(\mathbf{x}+\mathbf{x}')/2$.

The system then uses the bounds on $g_i(\mathbf{X})$ $(i=1,\ldots,n)$ obtained in step 710 to apply a linear method, that uses a Taylor expansion to linearize $f(\mathbf{x})$, to the relation $f(\mathbf{x}) \leq f\_bar$ (step 712).

Next, the system repeats step 707 (step 713).

Then, if the current box **X** is a sufficiently reduced version of the box $\mathbf{X}^{(1)}$ defined in step 703, the system puts **X** in list L_1 and goes to 703 (step 714).

If $N^H=4$, the system goes to step 720 (step 715).

Next, the system applies term consistency to the relation $H_{ii}(\mathbf{X})\geq 0$ for $i=1,\ldots,n$. If the result is empty, the system goes to step 703 (step 716).

If the box is changed in step 716, the system sets $N^H=0$ and goes to step 718. Otherwise, if the box is unchanged in step 716, the system replaces N^H by N^H+1 and goes to step 710 (step 717).

Next, the system repeats step 707 (step 718).

It also repeats step 714 (step 719).

The system then repeats steps 706-719 using box consistency in place of term consistency. However, the system skips steps 711-714. In step 710, the system omits the process of obtaining bounds on $g_i(\mathbf{X})$ (step 720).

If $w(\mathbf{X}) > (w_I+w_R)/2$, the system goes to step 732 (step 721). Note that, w_I denotes the width of the smallest box for which $\mathbf{M}^I = \mathbf{BJ}(\mathbf{x},\mathbf{X})$ is irregular. If $\mathbf{M}^I$ is regular for a given box, w_R denotes the width of the largest box for which $\mathbf{M}^I$ is regular. Initially, we set $w_I = w(\mathbf{X}^{(0)})$ and $w_R = 0$.

Next, the system computes the Jacobian $\mathbf{J}(\mathbf{x},\mathbf{X})$ of the gradient g. It also computes an approximate inverse $\mathbf{B}$ of the approximate center of $\mathbf{J}(\mathbf{x},\mathbf{X})$ and the matrix $\mathbf{M}(\mathbf{x},\mathbf{X})=\mathbf{BJ}(\mathbf{x},\mathbf{X})$ (step 722).

If the matrix $\mathbf{M}(\mathbf{x},\mathbf{X})$ is regular, the system finds the hull of the solution set of the linear system determined in step 722. If $\mathbf{M}(\mathbf{x},\mathbf{X})$ is irregular, the system applies one pass of the Gauss-Seidel method to the linear system. The system also updates w_I or w_R. If the result of the Newton step is empty, the system goes to step 703. If the interval Newton step proves the existence of a solution in $\mathbf{X}$, the system records this information (step 723).

Next, the system uses the gradient value $\mathbf{g}(\mathbf{x})$ and the matrix $\mathbf{B}$ computed in step 722 to compute the point $\mathbf{y}=\mathbf{x}-\mathbf{Bg}(\mathbf{x})$. The system uses the value of $f(\mathbf{y})$ to update *f_bar* (step 724).

The system then uses a quadratic method to solve $f(\mathbf{x}) \leq$ *f_bar* (step 725).

Next, the system repeats step 707 (step 726).

It also repeats step 714 (step 727).

Using the matrix $\mathbf{B}$ computed in step 722, the system analytically determines the system $\mathbf{Bg}(\mathbf{x})$, and then applies term consistency to solve the i-th

component of **Bg(x)** for the i-th variable x_i for $i=1,...,n$. If this procedure proves the existence of a solution in **X**, the system records this information (step 728).

Next, the system repeats step 707 (step 729).

It also repeats step 714 (step 730).

Next, the system applies box consistency to solve the i-th component of **Bg(x)** as determined in step 725 for the i-th variable for $i=1,...,n$. (step 731).

Next, the system repeats step 707 (step 732).

If the width of the box was reduced by a factor of at least eight in the Newton step in step 723, the system goes to step 722 (step 733).

Next, the system repeats step 714 (step 734).

The system then merges any overlapping gaps in components of **X** if any were generated using term consistency, box consistency, and/or the Newton method. If there are any gaps (after merging) are suitable for use in splitting **X**, the system goes to step 733. Suitability for splitting can be determined by ensuring that the splitting results in a sufficient size reduction in a component of **X**. If there are no such suitable gaps, the system goes to step 737 (step 735).

Next, the system selects three (or fewer if fewer exist) gaps and splits X into subboxes. The system then goes to step 703 (step 736).

The system then splits the three (or n if $n<3$) components of X for which D_i $(i=1,...,n)$ is largest. The system splits each such component by using a technique that gives preference to components over which the objective function varies the most. For example, we can approximate the change in f resulting from the i-th variable changing over X_i using the approximation $D_i = w[g_i(X)]w(X_i)$. The system then goes to step 703 (step 737).

Next, the system deletes any box **X** from list L_2 for which the lower bound of $f(\mathbf{X}) > f\_bar$. The system denotes the remaining boxes by $\mathbf{X}^{(1)},...,\mathbf{X}^{(s)}$ where s is the number of boxes remaining. The system determines the lower bound for the

global minimum f^* as the minimum over $i=1 \le i \le n$ of the lower bound of $f(\mathbf{X}^{(i)})$ (step 738).

Finally, the system terminates (step 739).

Discussion of the Process

The above-described process begins with procedures that involve the least amount of computing. We use term consistency first because it does not require computation of derivatives (and because it is effective in reducing large boxes).

For efficiency, the best box **X** to process is the one for which the lower bound of $f(\mathbf{X})$ is smallest. This tends to quickly reduce the upper bound *f_bar*. Because it is important to reduce *f_bar* as quickly as possible, the process returns to step 703 frequently to determine the box to process next.

We stop using the relations $H_n(\mathbf{x}) \ge 0$ $(i=1,...,n)$ when there is evidence that the remaining boxes are in a region where f is convex. See step 715. Using a similar philosophy, we begin using the Newton method more often when there is evidence that it will be effective. See step 721.

Note that the Jacobian $\mathbf{J}(\mathbf{x},\mathbf{X})$ is the Hessian of the objective function f. Therefore, knowing $\mathbf{J}(\mathbf{x},\mathbf{X})$ provides the means for determining the second order Taylor expansion of f. The gradient $\mathbf{g}(\mathbf{x})$ needed for this expansion is also computed when applying the Newton method in step 723.

Therefore, we have the data required to use the quadratic method to solve the relation $f(\mathbf{x}) \le f\_bar$. See step 725. If these data are not already available, it may not be worth generating them simply to solve $f(\mathbf{x}) \le f\_bar$ using the quadratic method.

Step 717 is done after term consistency is applied to the relation $H_n(\mathbf{X}) \ge 0$. When the box is unchanged, the count N^H is increased by 1. Step 717 is repeated

after box consistency is applied to what may be the same box. See step 720. Therefore, the count N^H can increase faster than one might otherwise expect.

The algorithm avoids too many applications of term consistency before changing procedures. Step 705 can force application of term consistency to the gradient instead of to the inequality $f \leq f\_bar$. Step 709 can force change from applying term consistency to the gradient to applying a Newton step. This takes precedence over our desire not to apply a Newton step when it may not be efficient.

We need occasional checks of efficiency of the Newton method because the current box may have become so small that the Newton method exhibits quadratic convergence and thus is more efficient than term consistency. When we force a change from term consistency, we also force a change from box consistency. This occurs because we do not apply the latter without having previously applied the former.

Step 733 causes the Newton step to be repeated. If the Newton method is exhibiting quadratic convergence, we want to take advantage of its ability to progress rapidly.

The foregoing descriptions of embodiments of the present invention have been presented only for purposes of illustration and description. They are not intended to be exhaustive or to limit the present invention to the forms disclosed. Accordingly, many modifications and variations will be apparent to practitioners skilled in the art. Additionally, the above disclosure is not intended to limit the present invention. The scope of the present invention is defined by the appended claims.